**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «ОРЕНБУРГСКИЙ ГОСУДАРСТВЕННЫЙ АГРАРНЫЙ УНИВЕРСИТЕТ»**

# МЕТОДИЧЕСКИЕ УКАЗАНИЯ ДЛЯ ОБУЧАЮЩИХСЯ ПО ОСВОЕНИЮ ДИСЦИПЛИНЫ

## Б1.Б.15 СУБД и базы данных

**Направление подготовки (специальность)** *09.03.01 "Информатика и вычислительная техника"*

**Профиль образовательной программы** *"Автоматизированные системы обработки информации и управления"*

**Форма обучения** *заочная*

**СОДЕРЖАНИЕ**

## 1. КОНСПЕКТ ЛЕКЦИЙ

### 1.1 Лекция № 1 (2 часа).

**Тема: «Введение в базы данных»**

#### 1.1.1 Вопросы лекции:

1. Понятие, назначение баз данных.
2. Основные компоненты баз данных.

#### 1.1.2 Краткое содержание вопросов:

**1. Понятие, назначение баз данных.**

В современных базах данных хранятся не только данные, но и информация. **База данных** (БД) – организованная структура, предназначенная для хранения информации. Современные БД позволяют размещать в своих структурах не только данные, но и методы (т.е. программный код), с помощью которых происходит взаимодействие с потребителем или другими программно-аппаратными комплексами.

**Системы управления базами данных** (СУБД) – комплекс программных средств, предназначенных для создания структуры новой базы, наполнения ее содержанием, редактирования содержимого и визуализации информации. Под **визуализацией информации базы** понимается отбор отображаемых данных в соответствии с заданным критерием, их упорядочение, оформление и последующая выдача на устройство вывода или передача по каналам связи.

Существует много систем управления базами данных. Они могут по-разному работать с разными объектами и предоставляют пользователю разные функции и средства. Большинство СУБД опираются на единый устоявшийся комплекс основных понятий.

**2. Основные компоненты баз данных.**

БД может содержать разные типы объектов. Каждая СУБД может реализовывать свои типы объектов.

**Таблицы** – основные объекты любой БД, в которых хранятся все данные, имеющиеся в базе, и хранится сама структура базы (поля, их типы и свойства).

**Отчеты** – предназначены для вывода данных, причем для вывода не на экран, а на печатающее устройство (принтер). В них приняты специальные меры для группирования выводимых данных и для вывода специальных элементов оформления, характерных для печатных документов (верхний и нижний колонтитулы, номера страниц, время создания отчета и другое).

**Страницы** или страницы доступа к данным – специальные объекты БД, выполненные в коде HTML, размещаемые на web -странице и передаваемые клиенту

вместе с ней. Сам по себе объект не является БД, посетитель может с ее помощью просматривать записи базы в полях страницы доступа. Т.о., страницы – интерфейс между клиентом, сервером и базой данных, размещенным на сервере.

**Макросы и модули** – предназначены для автоматизации повторяющихся операций при работе с системой управления БД, так и для создания новых функций путем программирования. Макросы состоят из последовательности внутренних команд СУБД и являются одним из средств автоматизации работы с базой. Модули создаются средствами внешнего языка программирования. Это одно из средств, с помощью которых разработчик БД может заложить в нее нестандартные функциональные возможности, удовлетворить специфические требования заказчика, повысить быстродействие системы управления, уровень ее защищенности.

**Запросы** – служат для извлечения данных из таблиц и предоставления их пользователю в удобном виде. С их помощью выполняют отбор данных, их сортировку и фильтрацию. Можно выполнить преобразование данных по заданному алгоритму, создавать новые таблицы, выполнять автоматическое заполнение таблиц данными, импортированными из других источников, выполнять простейшие вычисления в таблицах и многое другое.

Особенность запросов состоит в том, что они черпают данные из базовых таблиц и создают на их основе временную *результирующую таблицу* (*моментальный снимок* ) – образ отобранных из базовых таблиц полей и записей. Работа с образом происходит быстрее и эффективнее, нежели с таблицами, хранящимися на жестком диске.

Обновление БД тоже можно осуществить посредством запроса. В базовые таблицы все данные вносятся в порядке поступления, т.е. они не упорядочены. Но по соответствующему запросу можно получить отсортированные и отфильтрованные нужным образом данные.

**Формы** – средства для ввода данных, предоставляющие пользователю необходимые для заполнения поля. В них можно разместить специальные элементы управления (счетчики, раскрывающиеся списки, переключатели, флажки и прочее) для автоматизации ввода. Пример, заполнение определенных полей бланка. При выводе данных с помощью форм можно применять специальные средства их оформления.

**1.2 Лекция № 2 (2 часа).**

**Тема: «Обзор современных систем управления базами данных»**

**1.2.1 Вопросы лекции:**

1. Определение СУБД

2. Современные системы управления базами данных.

**1.2.2 Краткое содержание вопросов:**

**1. Определение СУБД.**

База данных (БД) – набор логически взаимосвязанных данных, описывающий информационное состояние объектов в различных предметных областях и обрабатываемые компьютерной техникой.

Системой управления базами данных является программная и языковая среда для создания, управления и обработки информационных баз.

Назначение СУБД:

- работа с базами на внешней (диски, ленты и т. д.) и оперативной памяти;
- совместный доступ пользователей;
- контроль изменений, архивирование и восстановление баз;
- предоставление языка доступа для обработки информации;
- утилиты для создания, модификации и управления базами.

**2. Современные системы управления базами данных.**

Технология “Клиент-сервер” – технология, разделяющая приложение- СУБД на две части: клиентскую (интерактивный графический интерфейс, расположенный на компьютере пользователя) и сервер, собственно осуществляющий управление данными, разделение информации, администрирование и безопасность, находящийся на выделенном компьютере. Взаимодействие “клиент-сервер” осуществляется следующим образом: клиентская часть приложения формирует запрос к серверу баз данных, на котором выполняются все команды, а результат исполнения запроса отправляется клиенту для просмотра и использования. Данная технология применяется, когда размеры баз данных велики, когда велики размеры вычислительной сети, и производительность при обработке данных, хранящихся не на компьютере пользователя (в крупном учреждении обычно имеет место именно такая ситуация). Если технология“клиент-сервер”на применяется, то для обработки даже нескольких записей весь файл копируется на компьютер пользователя, а только затем обрабатывается. При этом резко возрастает загрузка сети, и снижается производительность труда многих сотрудников.

MicrosoftAccess, MicrosoftVisualFoxPro, MicrosoftVisualBasic обеспечивают средства для создания клиентских частей в приложениях“клиент-сервер”, которые сочетают в себе средства просмотра, графический интерфейс и средства построения запросов, а Microsoft SQL Server является на сегодняшний день одним из самых мощных серверов баз данных.

OLE 2. 0 (ObjectLinkingandEmbedding – связывание и внедрение объектов) – стандарт, описывающий правила интеграции прикладных программ. Применяется для использования возможностей других приложений. OLE 2. 0 используется для определения и совместного использования объектов несколькими приложениями, которые поддерживают данную технологию. Например, использование в среде Access таблиц Excel и его мощных средств построения диаграмм или использование данных, подготовленных Access, в отчетах составленных в редакторе текстов Word (связывание или включение объекта).

OLE Automation (Автоматизация OLE) –компонент OLE, позволяющий программным путем устанавливать свойства и задавать команды для объектов другого приложения. Позволяет без необходимости выхода или перехода в другое окно использовать возможности нужного приложения. Приложение, позволяющее другим прикладным программам использовать свои объекты называетсяOLE сервером. Приложение, которое может управлять объектами OLE серверов называется OLE контроллер или OLE клиент. Из рассмотренных программных средств в качестве OLE серверов могут выступать MicrosoftAccess, а также MicrosoftExcel, Word и Graph.... MicrosoftVisualFoxPro 3. 0 и 5. 0 может выступать только в виде OLE клиента. RAD (RapidApplicationDevelopment – Быстрая разработка приложений) –подход к разработке приложений, предусматривающий широкое использование готовых компонентов и/или приложений и пакетов (в том числе от разных производителей). ODBC (OpenDatabaseConnectivity – открытый доступ к базам данных) –технология, позволяющая использовать базы данных, созданные другим приложением при помощи SQL.

SQL (StructuredQueryLanguage – язык структурированных запросов) – универсальный язык, предназначенный для создания и выполнения запросов, обработки данных как в собственной базе данных приложения, так и с базами данных, созданных другими приложениями, поддерживающими SQL. Также SQL применяется для управления реляционными базами данных.

VBA (VisualBasicforApplications – VisualBasic для Приложений) – разновидность объектно-ориентированного языка программирования VisualBasic, встраиваемая в программные пакеты.

**1. 3 Лекция № 3 (2 часа).**

**Тема: «Реляционная модель данных»**

**1.3.1 Вопросы лекции:**

1. Понятие домена, атрибута, кортежа, отношения.

2. Табличное представление отношения.

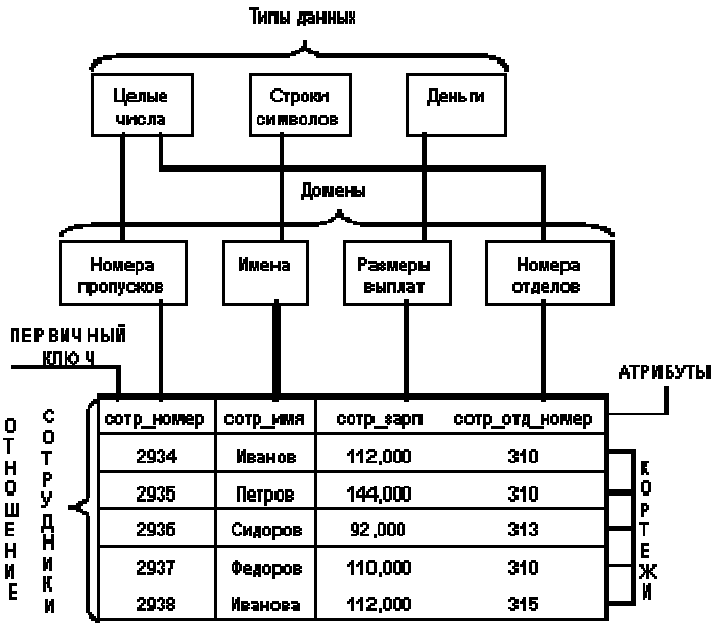
3. Схема отношения.

4. Первичные и внешние ключи

**1.3.2 Краткое содержание вопросов:**

**1. Понятие домена, атрибута, кортежа, отношения.**

Первые три относятся к элементам данных, остальные – к структурам, объединяющим элементы.




Для реляционной модели данных тип используемых данных не важен. Требование, чтобы тип данных был простым, нужно понимать так, что в реляционных операциях не должна учитываться внутренняя структура данных. Конечно, должны быть описаны действия, которые можно производить с данными как с единым целым, например, данные числового типа можно складывать, для строк возможна операция конкатенации и т.д.

В реляционной модели данных с понятием тип данных тесно связано понятие **домена**, которое можно считать уточнением типа данных.

Понятие домена более специфично для баз данных, хотя и имеет некоторые аналогии с подтипами в некоторых языках программирования. В самом общем виде домен определяется заданием некоторого базового типа данных, к которому относятся элементы домена, и произвольного логического выражения, применяемого к элементу типа данных. Если вычисление этого логического выражения дает результат "истина", то элемент данных является элементом домена.

Наиболее правильной интуитивной трактовкой понятия домена является понимание домена как допустимого потенциального множества значений данного типа.

Следует отметить также семантическую нагрузку понятия домена: данные считаются сравнимыми только в том случае, когда они относятся к одному домену. В нашем примере значения доменов "Номера пропусков" и "Номера групп" относятся к типу целых чисел, но не являются сравнимыми. Заметим, что в большинстве реляционных СУБД понятие домена не используется.

Домен характеризуется следующими свойствами:

- Домен имеет уникальное имя (в пределах базы данных).
- Домен определен на некотором простом типе данных или на другом домене.
- Домен может иметь некоторое логическое условие, позволяющее описать подмножество данных, допустимых для данного домена.

Домен несет определенную смысловую нагрузку.

Отличие домена от понятия подмножества состоит именно в том, что домен отражает семантику, определенную предметной областью. Может быть несколько доменов, совпадающих как подмножества, но несущие различный смысл. Например, домены "Вес детали" и "Имеющееся количество" можно одинаково описать как множество неотрицательных целых чисел, но смысл этих доменов будет различным, и это будут различные домены.

**Атрибут** есть имя, поставленное в соответствие домену и представляющее семантически значимое свойство объекта ПО. Если домену поставлено в соответствие имя, то говорят, что на домене определен атрибут. Атрибут принимает значения на домене.

На одном и том же домене можно определить произвольное число атрибутов. Атрибуты, определенные на общем домене, наследуют его свойства.

Отношение интуитивно можно понимать как таблицу, заголовком которой является строка атрибутов, а значимыми строками – строки их значений, или как плоский файл, однако это неточные представления.

Определение 3. Множество кортежей SR, соответствующих одной и той же схеме R, называется отношением.

Отношение характеризуется:

- арностью (степенью) – числом пар <домен, атрибут> в схеме;
- мощностью – числом кортежей, составляющих тело отношения.

Отношение является единственной структурной единицей РМД.

Кортеж, соответствующий данной схеме отношения, - это множество пар {имя атрибута, значение}, которое содержит одно вхождение каждого имени атрибута, принадлежащего схеме отношения. "Значение" является допустимым значением домена

данного атрибута (или типа данных, если понятие домена не поддерживается). Тем самым, степень или "арность" кортежа, т.е. число элементов в нем, совпадает с "арностью" соответствующей схемы отношения. Попросту говоря, кортеж - это набор именованных значений заданного типа.

Отношение - это множество кортежей, соответствующих одной схеме отношения. Иногда, чтобы не путаться, говорят "отношение-схема" и "отношение-экземпляр", иногда схему отношения называют заголовком отношения, а отношение как набор кортежей - телом отношения. На самом деле, понятие схемы отношения ближе всего к понятию структурного типа данных в языках программирования. Было бы вполне логично разрешать отдельно определять схему отношения, а затем одно или несколько отношений с данной схемой.

Свойства отношений РМД.

Отношения РМД обладают рядом свойств, отличающих их как от обычных теоретико-множественных отношений, так и от таблиц.

*Уникальность кортежей*. Так как отношение есть множество кортежей, в нем не может быть дубликатов кортежей, то есть каждый кортеж встречается в отношении только один раз. Из этого свойства следует, что каждое отношение имеет некоторый набор атрибутов, значения которых уникально идентифицирует кортежи. Этот набор атрибутов называют возможным ключом отношения.

*Неупорядоченность кортежей.* Это также следствие того, что отношение – множество кортежей. Множества неупорядоченны, если их упорядоченность специально не оговорена. Заметим, что в реальных структурах хранения данные так или иначе упорядочены. Однако учет этой упорядоченности в процедурах манипулирования данными сделал бы прикладные программы зависящими от физических структур хранения. Поэтому введение каких-либо гипотез об упорядоченности в концептуальную модель данных было бы ошибкой.

*Неупорядоченность атрибутов.* Это свойство следует из определения схемы отношения как множества пар <домен, атрибут>. Неупорядоченность атрибутов делает возможной модификацию схем отношений путем удаления атрибутов, вставки новых и переименования атрибутов и позволяет относительно просто определить ряд полезных операций над отношениями.

*Уникальность атрибутов.* Одноименные атрибуты недопустимы, поскольку это может привести к появлению в схеме отношения дубликатов пар (домен, атрибут), что противоречит определению. Кроме того, только уникальность атрибутов может обеспечить возможность отнесения значения из кортежа к определенному домену.

*Атомарность значений атрибутов.* Свойство следует из определения атрибута. Атрибут принимает значения на домене, а домен – подмножество простого типа. Таким образом, в реляционной теории не рассматриваются так называемые ненормализованные отношения.

Изменяемость отношений. Реляционная модель данных рассматривает отношение как структурный тип. Тип определяется схемой отношения. Все кортежи – знаки типа – удовлетворяют одной и той же схеме. Тело отношения может изменяться во времени. Отдельные кортежи могут добавляться или удаляться. Могут изменяться значения атрибутов в существующих кортежах. Поэтому можно говорить об экземпляре (текущем значении) отношения с заданной схемой.

Экземпляр (значение) отношения – это набор кортежей с заданной схемой, существующий в некоторый фиксированный момент времени.

**2. Табличное представление отношения.**

При табличном представлении каждому кортежу отношения взаимно-однозначно соответствует строка (запись) в таблице. Но не всякая таблица представляет некоторое отношение. Таблица — это прямоугольная сетка, в ячейках которой может находиться все, что угодно. При этом различные строки этой таблицы могут содержать в соответствующих столбцах одинаковые данные. Другими словами, таблицы могут содержать идентичные строки. В этом случае совокупность всех строк таблицы не является множеством, а значит, не представляет собой отношения. Напомню, что множество — это совокупность различных элементов. Таким образом, любое отношение (в определенном ранее смысле) можно представить в виде таблицы, но обратное утверждение, вообще говоря, не верно — не всякая таблица представляет какое-нибудь отношение.

При представлении отношения явным образом в виде таблицы, ее нельзя рассматривать просто как вместилище данных, в которое можно добавлять новые записи или удалять старые. Добавление, удаление и изменение данных в такой таблице приводит либо к другому отношению, либо к никакому отношению (если в таблице окажутся одинаковые записи).

Табличное представление отношений — это чрезвычайно плодотворный технологический прием, обеспечивший создание и широкое практическое применение баз данных. То обстоятельство, что отношения — просто множества, позволило изучать проблемы реляционных баз данных на теоретическом уровне, в мире математики (алгебра множеств и исчисление предикатов), а не только в мире технологии. В результате

разработка СУБД, конкретных баз данных, языков манипулирования данными, в том числе и 8<ЗЬ, водрузилась на твердый теоретический фундамент.

Далее мы будем рассматривать отношения, представляя их в виде таблиц. Введем несколько терминов, которые обычно применяются в теории отношений: Так, говоря о некотором отношении Я(А9 А29..., Ап), мы имеем в виду следующее.

□ Отношение имеет имя Л. Например, сведения о товарах, хранящихся на складе, образуют некоторое отношение, которому можно дать имя склад.

□ Множества А, А29..., Ап , для которых определено отношение Л, имеют различные имена, называемые атрибутами отношения. Мы будем считать, что А19 А29..., Ап — это атрибуты отношения. Совокупность всевозможных значений любого множества с именем А1 (/ = 1, 2,..., п) называется **доменом** атрибута Аг. Заметим еще раз, что в отношении не может быть двух и более одинаковых атрибутов, хотя их домены могут быть и одинаковыми (в смысле равенства множеств). Например, отношение склад может быть определено для атрибутов наименование, количество, цена и поставщик. Структуру этого отношения можно записать так: склад (наименование, количество, цена, поставщик). Каждый атрибут имеет свой домен — множество возможных значений. Например, доменом атрибута количество может быть множество неотрицательных чисел. При табличном представлении отношения каждому атрибуту соответствует заголовок столбца, а домену — множество значений в этом столбце.

□ Порядок перечисления атрибутов в структуре отношения не имеет значения. Иначе говоря, перестановка атрибутов местами оставляет само отношение прежним, хотя вид таблицы, представляющей это отношение, меняется. Например, отношения склад (наименование, количество, цена, поставщик) и склад (поставщик, наименование, количество, цена) считаются эквивалентными.

□ Элементами отношения являются кортежи — последовательности значений атрибутов отношения. В отношении не может быть двух и более одинаковых кортежей, а порядок расположения кортежей не имеет значения. При табличном представлении отношения каждому кортежу соответствует строка таблицы. Строки таблицы мы будем называть записями.

**3. Схема отношения**

Схема отношения базы данных — это именованное множество пар {имя атрибута, имя домена (или типа, если понятие домена не поддерживается)}. Если все атрибуты одного отношения определены на разных доменах, осмысленно использовать для именования атрибутов имена соответствующих доменов (не забывая, конечно, о том, что

это является всего лишь удобным способом именования и не устраняет различия между понятиями домена и атрибута).

Схема базы данных (в структурном смысле) — это набор именованных схем отношений.

Кортеж, соответствующий данной схеме отношения в базе данных, — это множество пар {имя атрибута, значение}, которое содержит одно вхождение каждого имени атрибута, принадлежащего схеме отношения. «Значение» является допустимым значением домена данного атрибута (или типа данных, если понятие домена не поддерживается). Тем самым, степень или «арность» кортежа, т.е. число элементов в нем, совпадает с «арностью» соответствующей схемы отношения. Попросту говоря, кортеж — это набор именованных значений заданного типа.

Отношение — это множество кортежей данной базы данных, соответствующих одной схеме отношения. Иногда, чтобы не путаться, говорят «отношение-схема» и «отношение-экземпляр», иногда схему отношения называют заголовком отношения, а отношение как набор кортежей — телом отношения. На самом деле, понятие схемы отношения в базе данных ближе всего к понятию структурного типа данных в языках программирования.

Число атрибутов в отношении называют степенью (или -арностью ) отношения.

Мощность множества кортежей отношения называют мощностью отношения.

**4. Первичные и внешние ключи**

Первичный ключ (англ. *primarykey*) — в реляционной модели данных один из потенциальных ключей отношения, выбранный в качестве основного ключа (или ключа по умолчанию).

Если в отношении имеется единственный потенциальный ключ, он является и первичным ключом. Если потенциальных ключей несколько, один из них выбирается в качестве первичного, а другие называют «альтернативными».

С точки зрения теории все потенциальные ключи отношения эквивалентны, то есть обладают одинаковыми свойствами *уникальности* и *минимальности*. Однако в качестве первичного обычно выбирается тот из потенциальных ключей, который наиболее удобен для тех или иных практических целей, например для создания внешних ключей в других отношениях либо для создания кластерного индекса. Поэтому в качестве первичного ключа, как правило, выбирают тот, который имеет наименьший размер (физического хранения) и/или включает наименьшее количество атрибутов.

Другой критерий выбора первичного ключа — сохранение уникальности со временем. Всегда существует вероятность того, что некоторый потенциальный ключ

перестанет быть таковым в долговременной перспективе или при изменении требований к системе. Например, если номер студенческой группы включает последнюю цифру года поступления, то номера групп для идентификации групп уникальны только в течение 10 лет. Поэтому в качестве первичного ключа стараются выбирать такой потенциальный ключ, который с наибольшей вероятностью не утратит уникальность.

Внешний ключ (англ. *foreignkey*) — понятие теории реляционных баз данных, относящееся к ограничениям целостности базы данных.

Неформально выражаясь, *внешний ключ* представляет собой *подмножество атрибутов* некоторой переменной отношения $R_2$, значения которых должны совпадать со значениями некоторого потенциального ключа некоторой переменной отношения $R_1$.

Формальное определение. Пусть $R_1$ и $R_2$ — две переменные отношения, не обязательно различные. Внешним ключом $FK$ в $R_2$ является подмножество атрибутов переменной $R_2$ такое, что выполняются следующие требования:

1. В переменной отношения $R_1$ имеется потенциальный ключ $CK$ такой, что $FK$ и $CK$ совпадают с точностью до переименования атрибутов (то есть переименованием некоторого подмножества атрибутов $FK$ можно получить такое подмножество атрибутов $FK'$, что $FK'$ и $CK$ совпадают как по именами, так и по типам атрибутов).

2. В любой момент времени каждое значение $FK$ в текущем значении $R_2$ идентично значению $CK$ в некотором кортеже в текущем значении $R_1$. Иными словами, в каждый момент времени множество всех значений $FK$ в $R_2$ является (нестрогим) подмножеством значений $CK$ в $R_1$.

При этом для данного конкретного внешнего ключа $FK \rightarrow CK$ отношение $R_1$, содержащее потенциальный ключ, называют *главным*, *целевым*, или *родительским* отношением, а отношение $R_2$, содержащее внешний ключ, называют *подчинённым*, или *дочерним* отношением.

Поддержка внешних ключей также называется соблюдением ссылочной целостности. Реляционные СУБД поддерживают автоматический контроль ссылочной целостности.

### 1.4 Лекция № 4 (2 часа).

**Тема: «Физическая модель данных»**

#### 1.4.1 Вопросы лекции:

1. Виды моделей данных.

2. Физическая модель данных

**1.4.2 Краткое содержание вопросов:**

**1. Виды моделей данных**

Выделяют следующие модели:

- инфологическая модель;
- даталогическая модель;
- физическая модель.

Инфологическая модель данных (ИМД) создается по результатам проведения исследований предметной области. Она представляет собой концептуальную модель базы данных, не привязанную к какой-либо конкретной СУБД.

Даталогическая модель данных (ДМД) создается на базе ИМД и представляет собой описание предметной области с учетом используемой для создаваемой БД модели данных.

Физическая модель данных (ФМД) – это модель данных, описанная с помощью средств конкретной СУБД. ФМД строится на базе даталогической путем добавления особенностей конкретной СУБД.

**2. Физическая модель данных**

Физические модели баз данных определяют способы размещения данных в среде хранения и способы доступа к этим данным, которые поддерживаются на физическом уровне. Исторически первыми системами хранения и доступа были файловые структуры и системы управления файлами (СУФ), которые фактически являлись частью операционных систем. СУБД создавала над этими файловыми моделями спою надстройку, которая позволяла организовать всю совокупность файлов таким образом, чтобы она работала как единое целое и получала централизованное управление от СУБД. Однако непосредственный доступ осуществлялся на уровне файловых команд, которые СУБД использовала при манипулировании всеми файлами, составляющими хранимые данные одной или нескольких баз данных.

Однако механизмы буферизации и управления файловыми структурами не приспособлены для решения задач собственно СУБД, эти механизмы разрабатывались просто для традиционной обработки файлов, и с ростом объемов хранимых данных они стали неэффективными для использования СУБД. Тогда постепенно произошел переход от базовых файловых структур к непосредственному управлению размещением данных на внешних носителях самой СУБД. И пространство внешней памяти уже выходило из-под владения СУФ и управлялось непосредственно СУБД. При этом механизмы, применяемые в файловых системах, перешли во многом и в новые системы организации данных во внешней памяти, называемые чаще страничными системами хранения информации. Поэтому наш раздел, посвященный физическим моделям данных, мы начнем с обзора

файлов и файловых структур, используемых для организации физических моделей, применяемых в базах данных, а в конце ознакомимся с механизмами организации данных во внешней памяти, использующими страничный принцип организации.

**1.5 Лекция № 5 (2 часа).**

**Тема: «Общая характеристика баз знаний и экспертных систем»**

**1.5.1 Вопросы лекции:**

1. Базы знаний
2. Экспертные системы

**1.5.2 Краткое содержание вопросов:**

**1. Базы знаний.**

База знаний — это особого рода база данных, разработанная для оперирования знаниями (метаданными). База знаний содержит структурированную информацию, покрывающую некоторую область знаний, для использования кибернетическим устройством (или человеком) с конкретной целью. Современные базы знаний работают совместно с системами поиска информации, имеют классификационную структуру и формат представления знаний.

Полноценные базы знаний содержат в себе не только фактическую информацию, но и правила вывода, допускающие автоматические умозаключения о вновь вводимых фактах и, как следствие, осмысленную обработку информации. Область наук об искусственном интеллекте, изучающая базы знаний и методы работы со знаниями, называется инженерией знаний.

Иерархический способ представления в базе знаний набора понятий и их отношений называется онтологией. Онтологию некоторой области знаний вместе со сведениями о свойствах конкретных объектов также можно назвать базой знаний.

**2. Экспертные системы**

Экспертная система − это компьютерная программа, которая моделирует рассуждения человека-эксперта в некоторой определенной области и использует для этого базу знаний, содержащую факты и правила об этой области, и некоторую процедуру логического вывода. Экспертные системы предназначены для моделирования и имитации логики опытных специалистов при принятии решения по какому-либо узкому вопросу в определенной предметной области.

ЭС можно считать одним из применений искусственного интеллекта, хотя исследователи искусственного интеллекта не ставили целью построение экспертных

систем. ЭС помогают специалистам, когда их собственных знаний, опыта и интуиции недостаточно для самостоятельного решения возникающих проблем. Такие системы представляют собой машинные программы, решающие задачи примерно так же, как решает их эксперт в реальной обстановке. Это позволяет накапливать, систематизировать и использовать знания и профессиональный опыт тех экспертов, которые выполняют конкретные задачи наилучшим образом и в первую очередь в тех областях, где задачи и их решения слабо формализованы или совсем не формализованы.

Типичная экспертная система состоит из следующих основных компонентов: инженер знаний, подсистема приобретения и накопления знаний, база знаний, рабочая область (база данных − БД), создатель заключения (решатель, интерпретатор), подсистема пояснений, интерфейс пользователя, пользователь.

Чтобы спроектировать экспертную систему, специалист, называемой инженером знания (специально подготовленный системный аналитик), тесно работает с одним или большим количеством экспертов в изучаемой области. Инженеры знания пытаются узнавать все относительно способа, которым эксперт принимает решения.

Подсистема приобретения и накопления знаний помогает инженеру знания в регистрации правил заключения и параметров в базе знаний. Знание (правила заключения и параметры), полученное инженером знания, затем с помощью подсистемы приобретения и накопления знаний регистрируется в компьютерной системе в специализированном формате в блоке, названном базой знаний.

База знаний в экспертной системе предназначена для хранения: долгосрочных данных, описывающих рассматриваемую область; правил, описывающих целесообразные преобразования данных этой области; заключений.

Рабочая область (база данных) предназначена для хранения исходных и промежуточных данных решаемой в текущий момент задачи.

Создатель заключения (решатель, интерпретатор), используя исходные данные из рабочей памяти и знания из базы знаний, формирует такую последовательность правил, которая будучи применима к исходным данным, приводит к решению задачи. Один и тот же создатель заключения может использоваться в различных экспертных системах с различной базой знаний.

Подсистема пояснений объясняет, как система получила решения задачи (или почему она не получила решения) и какие знания она при этом использовала, что облегчает эксперту-пользователю тестирование системы и повышает доверие пользователя к полученному результату.

Интерфейс пользователя (диалоговый компонент) ориентирован на организацию дружелюбного общения со всеми категориями пользователей, как в ходе решения задач, так и во время приобретения знаний, объяснения результатов работы.

В разработке экспертной системы участвуют представители следующих специальностей:

1. эксперт в той проблемной области, задачи которой будет решать ЭС;
2. инженер по знаниям, т.е. специалист по разработке ЭС;
3. программист, т.е. специалист по разработке инструментальных средств.

Необходимо отметить, что отсутствие среди участников разработки инженера по знаниям (т.е. замена его программистом) либо приводит к неудаче процесс создания ЭС, либо значительно удлиняет его.

Эксперт определяет знания (данные и правила), характеризирующие проблемную область, обеспечивает полноту и правильность введенных в ЭС знаний.

Инженер по знаниям помогает эксперту выявить и структурировать знания, необходимые для работы ЭС, осуществляет выбор того инструментального средства, которое наиболее подходит для данной проблемной области, и определяет способ предоставления знаний в этом ЭС. Он выделяет и программирует (традиционными средствами) стандартные функции (типичные для данной проблемной области), которые будут использоваться в правилах, вводимых экспертом.

Программист разрабатывает инструментальное средство, содержащее в пределе все основные компоненты ЭС, сопрягает инструментальное средство с той средой, в которой оно будет использовано.

## 2. МЕТОДИЧЕСКИЕ МАТЕРИАЛЫ ПО ВЫПОЛНЕНИЮ ЛАБОРАТОРНЫХ РАБОТ

**2.1 Лабораторная работа №1 (2 часа).**

**Тема: «Обзор современных систем управления базами данных»**

**2.1.1 Цель работы:** провести обзор современных СУБД.

**2.1.2 Задачи работы:**

1. Современные системы управления базами данных.

**2.1.3 Описание (ход) работы:**

Современные СУБД в основном являются приложениями Windows, так как данная среда позволяет более полно использовать возможности персональной ЭВМ, нежели среда DOS. Снижение стоимости высокопроизводительных ПК обусловил не только широкий переход к среде Windows, где разработчик программного обеспечения может в меньшей степени заботиться о распределении ресурсов, но также сделал программное обеспечение ПК в целом и СУБД в частности менее критичными к аппаратным ресурсам ЭВМ.

Среди наиболее ярких представителей систем управления базами данных можно отметить: LotusApproach, Microsoft Access, BorlanddBase, BorlandParadox, MicrosoftVisualFoxPro, MicrosoftVisualBasic, а также СУБД Microsoft SQL Server и Oracle, используемые в приложениях, построенных по технологии "клиент-сервер". Фактически, у любой современной СУБД существует аналог, выпускаемый другой компанией, имеющий аналогичную область применения и возможности, любое приложение способно работать со многими форматами представления данных, осуществлять экспорт и импорт данных благодаря наличию большого числа конвертеров. Общепринятыми, также, являются технологи, позволяющие использовать возможности других приложений, например, текстовых процессоров, пакетов построения графиков и т.п., и встроенные версии языков высокого уровня (чаще - диалекты SQL и/или VBA) и средства визуального программирования интерфейсов разрабатываемых приложений. Поэтому уже не имеет существенного значения, на каком языке и на основе какого пакета написано конкретное приложение, и какой формат данных в нем используется. Более того, стандартом "де-факто" стала "быстрая разработка приложений" или RAD (от английского RapidApplicationDevelopment), основанная на широко декларируемом в литературе "открытом подходе", то есть необходимость и возможность использования различных прикладных программ и технологий для разработки более гибких и мощных систем обработки данных. Поэтому в одном ряду с "классическими" СУБД все чаще упоминаются языки программирования VisualBasic 4.0 и Visual C++, которые позволяют быстро создавать необходимые компоненты приложений, критичные по скорости работы, которые трудно, а иногда невозможно разработать средствами "классических" СУБД. Современный подход к управлению базами данных подразумевает также широкое использование технологии "клиент-сервер".

Таким образом, на сегодняшний день разработчик не связан рамками какого-либо конкретного пакета, а в зависимости от поставленной задачи может использовать самые разные приложения. Поэтому, более важным представляется общее направление развития СУБД и других средств разработки приложений в настоящее время.

**2.2 Лабораторная работа №2 (2 часа).**

**Тема: «Архитектура СУБД»**

**2.2.1 Цель работы:** Рассмотреть архитектуру СУБД.

**2.2.2 Задачи работы:**

1. Архитектура СУБД.

2. Рассмотреть уровни архитектуры.

**2.2.3 Описание (ход) работы:**

Существует три основных уровня архитектуры или три уровня описания элементов данных. Это внешний, концептуальный и внутренний уровни, которые формируют так называемую трёхуровневую архитектуру.

Внешний уровень - уровень, на котором данные воспринимаются пользователями, тогда как СУБД и операционная система воспринимают данные на внутреннем уровне.

Концептуальный уровень представления данных осуществляет отображение внешнего уровня на внутренний и обеспечивает требуемую независимость друг от друга.

**2.3 Лабораторная работа №3 (2 часа).**

**Тема: «Модели данных»**

**2.3.1 Цель работы:** изучить модели данных.

**2.3.2 Задачи работы:**

1.Понятие модели данных

2.Сетевая модель данных

3.Реляционная модель данных

4.Иерархическая модель данных

**2.3.3 Описание (ход) работы:**

Сетевой подход к организации данных является расширением иерархического подхода. В иерархических структурах запись-потомок должна иметь в точности одного предка; в сетевой структуре данных у потомка может иметься любое число предков.

Сетевая БД состоит из набора записей и набора связей между этими записями, а если говорить более точно, из набора экземпляров каждого типа из заданного в схеме БД набора типов записи и набора экземпляров каждого типа из заданного набора типов связи.

Тип связи определяется для двух типов записи: предка и потомка. Экземпляр типа связи состоит из одного экземпляра типа записи предка и упорядоченного набора

экземпляров типа записи потомка. Для данного типа связи *L* с типом записи предка *P* и типом записи потомка *C* должны выполняться следующие два условия:

- каждый экземпляр типа записи *P* является предком только в одном экземпляре типа связи *L*;
- каждый экземпляр типа записи *C* является потомком не более чем в одном экземпляре типа связи *L*.

На формирование типов связи не накладываются особые ограничения; возможны, например, следующие ситуации:

- тип записи потомка в одном типе связи *L1* может быть типом записи предка в другом типе связи *L2*(как в иерархии);
- данный тип записи *P* может быть типом записи предка в любом числе типов связи;
- данный тип записи *P* может быть типом записи потомка в любом числе типов связи;
- может существовать любое число типов связи с одним и тем же типом записи предка и одним и тем же типом записи потомка; и если *L1* и *L2* - два типа связи с одним и тем же типом записи предка *P* и одним и тем же типом записи потомка *C*, то правила, по которым образуется родство, в разных связях могут различаться;
- типы записи *X* и *Y* могут быть предком и потомком в одной связи и потомком и предком – в другой;
- предок и потомок могут быть одного типа записи.

Почти все современные системы основаны на **реляционной** (relational) модели управления базами данных. Название **реляционная** связано с тем, что каждая запись в такой базе данных содержит информацию, относящуюся только к одному конкретному объекту.

В **реляционной** СУБД все обрабатываемые данные представляются в виде плоских таблиц. Информация об объектах определенного вида представляется в табличном виде: в столбцах таблицы сосредоточены различные атрибуты объектов, а строки предназначены для сведения описаний всех атрибутов к отдельным экземплярам объектов.

Модель, созданная на этапе инфологического моделирования, в наибольшей степени удовлетворяет принципам реляционности. Однако для приведения этой модели к реляционной необходимо выполнить процедуру, называемую **нормализацией**.

Теория нормализации оперирует с пятью **нормальными формами**. Эти формы предназначены для уменьшения избыточности информации, поэтому каждая последующая нормальная форма должна удовлетворять требованиям предыдущей и

некоторым дополнительным условиям. При практическом проектировании баз данных четвертая и пятая формы, как правило, не используются. Мы ограничились рассмотрением первых четырех нормальных форм.

Введем понятия, необходимые для понимания процесса приведения модели к реляционной схеме.

**Отношение** - абстракция описываемого объекта как совокупность его свойств. Проводя инфологический этап проектирования, мы говорили об абстракции объектов и приписывали им некоторые свойства. Теперь же, проводя концептуальное проектирование, мы переходим к следующему уровню абстракции. На данном этапе объектов, как таковых, уже не существует. Мы оперируем совокупностью свойств, которые и определяют объект.

**Экземпляр отношения** - совокупность значений свойств конкретного объекта.

**Первичный ключ** - идентифицирующая совокупность атрибутов, т.е. значение этих атрибутов уникально в данном отношении. Не существует двух экземпляров отношения содержащих одинаковые значения в первичном ключе.

**Простой атрибут** - атрибут, значения которого неделимы.

**Сложный атрибут** - атрибут, значением которого является совокупность значений нескольких различных свойств объекта или несколько значений одного свойства.

В классической теории баз данных, модель данных есть формальная теория представления и обработки данных в системе управления базами данных(СУБД), которая включает, по меньшей мере, три аспекта:

– аспект структуры: методы описания типов и логических структур данных в базе данных;

– аспект манипуляции: методы манипулирования данными;

– аспект целостности: методы описания и поддержки целостности базы данных.

Аспект структуры определяет, что из себя логически представляет база данных, аспект манипуляции определяет способы перехода между состояниями базы данных (то есть способы модификации данных) и способы извлечения данных из базы данных, аспект целостности определяет средства описаний корректных состояний базы данных.

Модель данных — это абстрактное, самодостаточное, логическое определение объектов, операторов и прочих элементов, в совокупности составляющих абстрактную машину доступа к данным, с которой взаимодействует пользователь. Эти объекты позволяют моделировать структуру данных, а операторы — поведение данных[1].

Каждая БД и СУБД строится на основе некоторой явной или неявной модели данных. Все СУБД, построенные на одной и той же модели данных, относят к одному типу. Например, основой реляционных СУБД является реляционная модель данных, сетевых СУБД — сетевая модель данных, иерархических СУБД — иерархическая модель данных и т. д.

Иерархическая модель данных — это модель данных, где используется представление базы данных в виде древовидной (иерархической) структуры, состоящей из объектов (данных) различных уровней.

Между объектами существуют связи, каждый объект может включать в себя несколько объектов более низкого уровня. Такие объекты находятся в отношении предка (объект более близкий к корню) к потомку (объект более низкого уровня), при этом возможна ситуация, когда объект-предок не имеет потомков или имеет их несколько, тогда как у объекта-потомка обязательно только один предок. Объекты, имеющие общего предка, называются близнецами (в программировании применительно к структуре данных дерево устоялось название братья).

Базы данных с иерархической моделью одни из самых старых, и стали первыми системами управления базами данных для мейнфреймов. Разрабатывались в 1950-х и 1960-х, например, InformationManagementSystem (IMS) фирмы IBM.

**2.4 Лабораторная работа №4 (2 часа).**

**Тема: «Реляционная модель данных»**

**2.4.1 Цель работы:** рассмотреть реляционную модель данных.

**2.4.2 Задачи работы:**

1.Понятие домена, атрибута, кортежа, отношения

2. Табличное представление отношения

3.Схема отношения

**2.4.3 Описание (ход) работы:**

Понятие *домена* более специфично для баз данных, хотя и имеет некоторые аналогии с подтипами в некоторых языках программирования. В самом общем виде домен определяется заданием некоторого базового типа данных, к которому относятся элементы домена, и произвольного логического выражения, применяемого к элементу типа данных. Если вычисление этого логического выражения дает результат "истина", то элемент данных является элементом домена.

Кортеж, соответствующий данной схеме отношения, - это множество пар {имя атрибута, значение}, которое содержит одно вхождение каждого имени атрибута, принадлежащего схеме отношения. "Значение" является допустимым значением домена данного атрибута (или типа данных, если понятие домена не поддерживается). Тем самым, степень или "арность" кортежа, т.е. число элементов в нем, совпадает с "арностью" соответствующей схемы отношения. Попросту говоря, кортеж - это набор именованных значений заданного типа.

Отношение - это множество кортежей, соответствующих одной схеме отношения. Иногда, чтобы не путаться, говорят "отношение-схема" и "отношение-экземпляр", иногда схему отношения называют заголовком отношения, а отношение как набор кортежей - телом отношения. На самом деле, понятие схемы отношения ближе всего к понятию структурного типа данных в языках программирования. Было бы вполне логично разрешать отдельно определять схему отношения, а затем одно или несколько отношений с данной схемой.

Кортеж, соответствующий данной схеме отношения, - это множество пар {имя атрибута, значение}, которое содержит одно вхождение каждого имени атрибута, принадлежащего схеме отношения. Попросту говоря, кортеж - это набор именованных значений заданного типа.

Понятие *реляционный* (англ.*relation* -отношение) связано с разработками известного американского специалиста в области систем баз данных Е. Кодда.

Эти модели характеризуются простотой структуры данных, удобным для пользователя табличным представлением и возможностью использования формального аппарата алгебры отношений и реляционного исчисления для обработки данных.

Реляционная модель ориентирована на организацию данных в виде двумерных таблиц. Каждая ***реляционная таблица*** представляет собой двумерный массив и обладает следующими свойствами:

- каждый элемент таблицы - один элемент данных;
- все столбцы в таблице однородные, т.е. все элементы в столбце имеют одинаковый тип (числовой, символьный и т.д.) и длину;
- каждый столбец имеет уникальное имя;
- одинаковые строки в таблице отсутствуют;
- порядок следования строк и столбцов может быть произвольным.

**Пример.** Реляционной таблицей можно представить информацию о студентах, обучающихся в вузе (рис. 7).

| N личного дела | Фамилия | Имя | Отчество | Дата рождения | Группа |
|---|---|---|---|---|---|
| | Сергеев | Петр | Михайлович | 01.01.76 | |
| | Петрова | Анна | Владимировна | 15.03.75 | |
| | Анохин | Андрей | Борисович | 14.04.76 | |

**Рис. 7.** Пример реляционной таблицы

Отношения представлены в виде *таблиц,* строки которых соответствуют *записям,* а столбцы - *полям.*

Поле, каждое значение которого однозначно определяет соответствующую запись, называется ***простым ключом*** (ключевым полем). Если записи однозначно определяются значениями нескольких полей, то такая таблица базы данных имеет***составной ключ***. В примере, показанном на рис. 7, ключевым полем таблицы является "N личного дела".

Чтобы связать две реляционные таблицы, необходимо ключ первой таблицы ввести в состав ключа второй таблицы (возможно совпадение ключей); в противном случае нужно ввести в структуру первой таблицы *внешний ключ* - ключ второй таблицы.

***Схемой отношения*** называется перечень имен атрибутов данного отношения с указанием домена, к которому они относятся:

$$S_R = (A_1, A_2, A_n),\ A_i \subseteq D_i.$$

Если атрибуты принимают значения из одного и того же домена, то они называются $\theta$-сравнимыми, где $\theta$ – множество допустимых операций сравнений, заданных для данного домена. Например, если домен содержит числовые данные, то для него допустимы все операции сравнения, тогда $\theta=\{=, <>, >=, <=, <, >\}$. Однако, и для доменов, содержащих символьные данные, могут быть заданы не только операции сравнения по равенству и неравенству значений. Если для данного домена задано лексикографическое упорядочение, то он имеет также полный спектр операций сравнения.

Схемы двух отношений называются ***эквивалентными***, если они имеют одинаковую степень и возможно такое упорядочение имен атрибутов в схемах, что на одинаковых местах будут находиться сравнимые атрибуты, то есть атрибуты, принимающие значения из одного домена:

Пусть $S_{R_1} = (A_1, A_2, \ldots, A_n)$ – схема отношения $R_1$. $S_{R_2} = (B_{i1}, B_{i2}, \ldots, B_{in})$ – схема отношения $R_2$ после упорядочения имен атрибутов. Тогда

$$S_{R_1} \sim^{S_{R_2}} \Leftrightarrow \begin{cases} 1.\ n = m, \\ 2.\ A_j, B_{ij} \subseteq D_j. \end{cases}$$

Таким образом, для эквивалентных отношений выполняются следующие условия:

- Таблицы имеют одинаковое количество столбцов.
- Таблицы содержат столбцы с одинаковыми наименованиями.
- Столбцы с одинаковыми наименованиями содержат данные из одних и тех же доменов.
- Таблицы имеют одинаковые строки с учетом того, что порядок столбцов может различаться.

Все такие таблицы есть различные *изображения* одного и того же отношения.

**2.5 Лабораторная работа №5 (2 часа).**

**Тема: «Реляционная алгебра и язык SQL»**

**2.5.1 Цель работы:** изучить реляционную алгебру и язык SQL.

**2.5.2 Задачи работы:**

1. Особенности языков описания и манипулирования данными в реляционной модели

2. языки запросов, основанные на реляционном исчислении.

3. структурный язык запросов SQL.

**2.5.3 Описание (ход) работы:**

Двумя фундаментальными языками запросов к реляционным БД являются языки реляционной алгебры и реляционного исчисления. При всей своей строгости и теоретической обоснованности эти языки редко используются в современных реляционных СУБД в качестве средств пользовательского интерфейса. Запросы на этих языках трудно формулировать и понимать. SQL представляет собой некоторую комбинацию реляционного исчисления кортежей и реляционной алгебры, причем до сих пор нет общего согласия, к какому из классических языков он ближе. При этом возможности SQL шире, чем у этих базовых реляционных языков, в частности, в общем случае невозможна трансляция запроса, сформулированного на SQL, в выражение реляционной алгебры, требуется некоторое ее расширение.

Существенными свойствами подъязыка запросов SQL являются возможность простого формулирования запросов с соединениями нескольких отношений и использование вложенных подзапросов в предикатах выборки. Вообще говоря,

одновременное наличие обоих средств избыточно, но это дает пользователю при формулировании запроса возможность выбора более понятного ему варианта.

В предикатах со вложенными подзапросами в SQL System R можно употреблять теретико-множественные операторы сравнения, что позволяет формулировать квантифицированные запросы (эти возможности обычно труднее всего понимаются пользователями и поэтому в дальнейшем в SQL появились явно квантифицируемые предикаты).

Существенной особенностью SQL является возможность указания в запросе потребности группирования отношения-результата по указанным полям с поддержкой условий выборки на всю группу целиком. Такие условия выборки могут содержать агрегатные функции, вычисляемые на группе. Эта возможность SQL главным образом отличает этот язык от языков реляционной алгебры и реляционного исчисления, не содержащих аналогичных средств.

Еще одним отличием SQL является необязательное удаление кортежей-дубликатов в окончательном или промежуточных отношениях-результатах. Строго говоря, результатом оператора выборки в языке SQL является не отношение, а мультимножество кортежей. В тех случаях, когда семантика запроса требует наличия отношения, уничтожение дубликатов производится неявно.

Самый общий вид запроса на языке SQL представляет теоретико-множественное алгебраическое выражение, составленное из элементарных запросов. В SQL System R допускались все базовые теретико-множественные операции (UNION, INTERSECT и MINUS).

Работа с неопределенными значениями в SQL System R до конца продумана не была, хотя неявно предполагалось использование трехзначной логики при вычислении логических выражений.

Операторы манипулирования данными UPDATE и DELETE построены на тех же принципах, что и оператор выборки данных SELECT. Набор кортежей указанного отношения, подлежащих модификации или удалению, определяется входящим всоответствующий оператор логическим выражением, которое может включать сложные предикаты, в том числе и с вложенными подзапросами.

В операторе вставки кортежа(ей) в указанное отношение заносимый кортеж может задаваться как в литеральной форме, так и с помощью внутреннего подоператора выборки.

В число операторов определения схемы БД SQL System R входили операторы создания и уничтожения постоянных и временных хранимых отношений (CREATE

TABLE и DROP TABLE) и создания и уничтожения представляемых отношений (CREATE VIEW и DROP VIEW). В языке и в реализации System R не запрещалось использовать операторы определения схемы в пределах транзакции, содержащей операторы выборки и манипулирования данными. Допускалось, например, использование операторов выборки и манипулирования данными, в которых указываются отношения, не существующие в БД к моменту компиляции оператора. Конечно, эта возможность существенно усложняла реализацию и требовалась по существу очень редко.

Оператор манипулирования схемой БД ALTER TABLE позволял добавлять указываемые поля к существующим отношениям. В описании языка определялось, что выполнение этого оператора не должно приводить к недействительности ранее откомпилированных операторов над отношением, схема которого изменяется, и что значения вновь определенных полей в существующих кортежах отношения становятся неопределенными.

Язык SQL System R включал очень мощные средства контроля и поддержания целостности БД. Средства контроля базировались на аппарате ограничений целостности (ASSERTIONS). Фактически, ограничение целостности - это логическое выражение, вычисляемое над текущим состоянием БД, ложность которого соответствует нецелостному состоянию БД. Логическое выражение ограничения целостности могло содержать любой допустимый в языке предикат.

Механизмы ограничений целостности и триггеров System R являлись очень мощными и общими, но реализация их очень трудна и накладна (как уже отмечалось, триггеры так и не были реализованы в System R). Дополнительную сложность в реализации создавал тот факт, что допускалось (по крайней мере не запрещалось языком) определение ограничений целостности и триггеров в пределах той же транзакции, в которой выполняются операторы манипулирования данными. При наиболее полной реализации требовалось бы большое число дополнительных действий во время выполнения транзакции. Кроме того, в ряде случаев отсутствие зафиксированной семантики соответствующих конструкций языка приводило к неоднозначному пониманию выполнения транзакций.

В языке отсутствуют какие-либо ограничения по поводу использования представлений: в любом операторе SQL, в котором допускается использование имени хранимого отношения, допускается и использование имени представления. В SQL System R ничего не говорится о рекомендуемом способе реализации доступа к представлениям, но при любом способе эффект должен быть таким, как если бы выполнить полную материализацию представления до выполнения оператора.

Внесение в реляционный язык, каким является SQL, явных операторов порождения и уничтожения структур физического уровня, поддерживающих эффективное выполнение запросов к БД, явилось в SQL System R чисто прагматическим решением, обеспечивающим возможность всех видов работ с БД с помощью одного языка.

В SQL System R упоминаются два вида таких структур: индексы и связи (links). Индекс в его абстрактном языковом представлении - это инвертированный файл, обеспечивающий доступ к кортежам соответствующего отношения на основе заданных значений одного или нескольких столбцов, составляющих ключ индекса. Операторы языка позволяли создавать и уничтожать индексы, но никаким образом не давали возможности явно указать на необходимость использования существующего индекса при выполнении оператора выборки, решение об этом возлагалось на реализацию.

С помощью оператора определения индекса можно было выразить два дополнительных утверждения, касающихся логической схемы отношения и физической структуры его хранения. Использование при определении индекса ключевого слова UNIQUE означало, что ключ этого индекса является возможным ключом соответствующего отношения. Фактически это означает наличие дополнительного механизма определения ограничения целостности отношения. Один из индексов для данного отношения мог быть определен с ключевым словом CLUSTERING. Это означает требование физической кластеризации во внешней памяти кортежей отношения с равными или близкими значениями ключа индекса.

Операторы определения связи позволяли в стиле сетевой модели данных организовать во внешней памяти списки кортежей указанного отношения. Как и в случае индексов, операторы позволяли создавать и уничтожать такие списки, но не давали возможности явно указать на необходимость использования существующих списков при выполнении операторов выборки. Большая трудоемкость поддержания списков при выполнении операторов манипулирования данными и трудность выполнения оценок стоимости их использования при выполнении операторов выборки привели к тому, что механизм связей исчез из языка уже на поздней стадии проекта System R. С тех пор этот механизм, насколько нам известно, не появлялся ни в одном варианте SQL.

Существенной особенностью языка SQL, появившейся в нем с самого начала, является обеспечение защиты доступа к данным средствами самого языка. Основная идея такого подхода состоит в том, что по отношению к любому отношению БД и любому столбцу отношения вводится предопределенный набор привилегий. С каждой транзакцией неявно связывается идентификатор пользователя, от имени которого она выполняется

(способы связи и идентификации пользователей не фиксируются в языке и определяются в реализации).

После создания нового отношения все привилегии, связанные с этим отношением и всеми его столбцами, принадлежат только пользователю-создателю отношения. В число привилегий входит привилегия передачи всех или части привилегий другому пользователю, включая привилегию на передачу привилегий. Технически передача привилегий осуществляется при выполнении оператора SQL GRANT. Существует также привилегия изъятия всех или части привилегий у пользователя, которому они ранее были переданы. Эта привилегия также может передаваться. Технически изъятие привилегий происходит при выполнении оператора SQL REVOKE.

Проверка полномочности доступа к данным происходит на основе информации о полномочиях, существующих во время компиляции соответствующего оператора SQL. Подобно тому, что мы отмечали в связи с ограничениями целостности и триггерами, в SQL System R отсутствовали какие-либо ограничения по поводу использования операторов GRANT и REVOKE. Это приводило к существенным техническим затруднениям в реализации, а иногда к неоднозначному пониманию поведения.

Долгое время подход к защите данных от несанкционированного доступа принимался практически без критики, однако в связи с распространяющимся использованием реляционных СУБД в нетрадиционных приложениях все чаще раздается критика. Если, например, в системе БД должна поддерживаться многоуровневая защита данных, соответствующую систему полномочий весьма трудно, а иногда и невозможно построить на основе средств SQL.

В SQL System R существовали два специальных оператора для установки так называемых точек сохранения транзакции и для отката транзакции к ранее установленной точке сохранения. В литературе, относящейся к System R, обсуждение этих возможностей практически не содержится, из чего неявно следует, что они не были реализованы.

Прямолинейная реализация этого механизма не вызывает особых технических затруднений, но и не очень полезна, потому что после выполнения частичного отката транзакции для успешного продолжения работы прикладной программы потребовалось бы и восстановить ее состояние в соответствующей точке, а это никак не поддерживается. Понятно, что при более тщательной проработке должны быть увязаны механизмы точек сохранения и контроля целостности. Например, было бы естественно, чтобы при выполнении оператора ENFORCE INTEGRITY, если какие-либо ограничения целостности нарушаются, происходил автоматический откат транзакции к ближайшей точки сохранения, в которой нарушения целостности БД не было. Это значительно усложнило

бы реализацию, но было бы очень полезно. Аналогично, можно было бы использовать механизм точек сохранения при автоматических откатах транзакций по причине возникновения синхронизационных тупиков.

Отметим еще два важных свойства языка SQL System R, которые в разных видах присутствуют во всех развитых последующих вариантах языка.

В SQL System R присутствуют специальные операторы, поддерживающие встраивание операторов SQL в традиционные языки программирования (в System R основным таким языком был PL/1).

Основная проблема встраивания SQL в язык программирования состояла в том, что SQL - реляционный язык, т.е. его операторы большей частью работают со множествами, в то время как в языках программирования основными являются скалярные операции. Решение SQL состоит в том, что в язык дополнительно включаются операторы, обеспечивающие покортежный доступ к результату запроса к БД.

Для этого в язык вводится понятие курсора, с которым связывается оператор выборки. Над определенным курсором можно выполнять оператор OPEN, означающий материализацию отношения-результата запроса, оператор FETCH, позволяющий выбрать очередной кортеж результирующего отношения в память программы, и оператор CLOSE, означающий конец работы с данным курсором.

Дополнительную гибкость при создании прикладных программ со встроенным SQL обеспечивает возможность параметризации операторов SQL значениями переменных включающей программы.

Для упрощения создания интерактивных SQL-ориентированных систем в SQL System R были включены операторы, позволяющие во время выполнения транзакции откомпилировать и выполнить любой оператор SQL.

Оператор PREPARE вызывает динамическую компиляцию оператора SQL, текст которого содержится в указанной переменной символьной строке включающей программы. Текст может быть помещен в переменную при выполнении программы любым допустимым способом, например, введен с терминала.

Оператор DESCRIBE служит для получения информации об указанном операторе SQL, ранее подготовленном с помощью оператора PREPARE. С помощью этого оператора можно узнать, во-первых, является ли подготовленный оператор оператором выборки, и во-вторых, если это оператор выборки, получить полную информацию о числе и типах столбцов результирующего отношения.

Для выполнения ранее подготовленного оператора SQL, не являющегося оператором выборки, служит оператор EXECUTE. Для выполнения динамически

подготовленного оператора выборки используется аппарат курсоров с некоторыми отличиями по части задания адресов переменных включающей программы, в которые должны быть помещены значения столбцов текущего кортежа результата.

Подводя итог приведенному краткому описанию основных черт SQL System R, отметим, что несмотря на недостаточную техническую проработку, в идейном отношении язык содержал все необходимые средства, позволяющие использовать его как базовый язык СУБД.

**2.6 Лабораторная работа №6 (2 часа).**

**Тема: «Проектирование концептуальной модели данных»**

**2.6.1 Цель работы:** рассмотреть проектирование концептуальной модели данных.

**2.6.2 Задачи работы:**

1. Анализ данных

2. Нормализация отношений

**2.6.3 Описание (ход) работы:**

В качестве примера возьмем базу данных компании, которая занимается издательской деятельностью. База данных создаётся для информационного обслуживания редакторов, менеджеров и других сотрудников компании. БД должна содержать данные о сотрудниках компании, книгах, авторах, финансовом состоянии компании и предоставлять возможность получать разнообразные отчёты. В соответствии с предметной областью система строится с учётом следующих особенностей: каждая книга издаётся в рамках контракта; книга может быть написана несколькими авторами; контракт подписывается одним менеджером и всеми авторами книги; каждый автор может написать несколько книг (по разным контрактам); порядок, в котором авторы указаны на обложке, влияет на размер гонорара; если сотрудник является редактором, то он может работать одновременно над несколькими книгами; у каждой книги может быть несколько редакторов, один из них – ответственный редактор; каждый заказ оформляется на одного заказчика; в заказе на покупку может быть перечислено несколько книг. Выделим базовые сущности этой предметной области: Сотрудники компании. Атрибуты сотрудников – ФИО, табельный номер, пол, дата рождения, паспортные данные, ИНН, должность, оклад, домашний адрес и телефоны. Для редакторов необходимо хранить сведения о редактируемых книгах; для менеджеров – сведения о подписанных контрактах. Авторы. Атрибуты авторов – ФИО, ИНН (индивидуальный номер налогоплательщика), паспортные данные, домашний адрес, телефоны. Для авторов необходимо хранить

сведения о написанных книгах. Книги. Атрибуты книги – авторы, название, тираж, дата выхода, цена одного экземпляра, общие затраты на издание, авторский гонорар. Контракты будем рассматривать как связь между авторами, книгами и менеджерами. Атрибуты контракта – номер, дата подписания и участники. Для отражения финансового положения компании в системе нужно учитывать заказы на книги. Для заказа необходимо хранить номер заказа, заказчика, адрес заказчика, дату поступления заказа, дату его выполнения, список заказанных книг с указанием количества экземпляров.

Нормализация отношений - обеспечивает эффективность структур данных в реляционной БД

Этот процесс уменьшает избыточность данных (хранение одинаковых данных в нескольких местах). В результате более рационально используется внешняя память, уменьшается вероятность нарушения согласованности данных.

Нормализация представляет собой действия по последовательному преобразованию исходной (ненормализованной) таблицы в нормализованные отношения в первой нормальной форме (1НФ), 2НФ, 3НФ, нормальной форме Бойса-Кодда (НФБК), 4НФ, 5НФ.

На практике, как правило, ограничиваются 3НФ, ее оказывается вполне достаточно для создания надежной схемы БД.

Основные свойства нормальных форм:

- каждая следующая нормальная форма улучшает свойства предыдущей нормальной формы;

- при переходе к следующей нормальной форме свойства предыдущих нормальных форм сохраняются.

При проектировании баз данных упор в первую очередь делается на достоверность и непротиворечивость хранимых данных, причем эти свойства не должны утрачиваться в процессе работы с данными, т.е. после многочисленных изменений, удалений и дополнений данных по отношению к первоначальному состоянию БД.

Для поддержания БД в устойчивом состоянии используется ряд механизмов, которые получили обобщенное название средств поддержки целостности. Эти механизмы применяются как статически (на этапе проектирования БД), так и динамически (в процессе работы с БД).Приведение структуры БД в соответствие этим ограничениям - это и есть нормализация.

В целом суть этих ограничений весьма проста: каждый факт, хранимый в БД, должен храниться один-единственный раз, поскольку дублирование может привести (и на практике непременно приводит, как только проект приобретает реальную сложность) к

несогласованности между копиями одной и той же информации. Следует избегать любых неоднозначностей, а также избыточности хранимой информации.

Нормализация отношений - обеспечивает эффективность структур данных в реляционной БД

Этот процесс уменьшает избыточность данных (хранение одинаковых данных в нескольких местах). В результате более рационально используется внешняя память, уменьшается вероятность нарушения согласованности данных.

**2.7 Лабораторная работа №7 (2 часа).**

**Тема: «Администрирование базы данных»**

**2.7.1 Цель работы:** изучить администрирование базы данных.

**2.7.2 Задачи работы:**

1. Функция администрирования базы данных
2. Жизненный цикл системы с базой данных

**2.7.3 Описание (ход) работы:**

1.Администрирование базы данных – это функция управления базой данных (БД). Лицо ответственное за администрирование БД называется “Администратор базы данных” (АБД) или “DatabaseAdministrator” (DBA).

Функция “администрирования данных” стала активно рассматриваться и определяться как вполне самостоятельная с конца 60-х годов. Практическое значение это имело для предприятий, использующих вычислительную технику в системах информационного обеспечения для своей ежедневной деятельности. Специализация этой функции с течением времени совершенствовалась, но качественные изменения в этой области стали происходить с началом использования так называемых интегрированных баз данных. Одна такая база данных могла использоваться для решения многих задач.

Таким образом, сформировалось определение БД как общего информационного ресурса предприятия, которое должно находиться всегда в работоспособном состоянии. И как для каждого общего ресурса значительной важности, БД стала требовать отдельного управления. Во многих случаях это было необходимо для обеспечения её повседневной эксплуатации, её развития в соответствии с растущими потребностями предприятия. К тому же БД и технология её разработки постоянно совершенствовались и уже требовались специальные знания высокого уровня для довольно сложного объекта, которым стала база данных. Отсюда функция управления базой данных и получила название

“Администрирование базы данных”, а лицо ею управляющее стали называть “Администратор баз данных”.

**2.8 Лабораторная работа №8 (2 часа).**

**Тема: «СУБД ACCESS»**

**2.8.1 Цель работы:** Изучить СУБД Access.

**2.8.2 Задачи работы:**

1. Назначение, общая характеристика и структура СУБД ACCESS

2. Состав БД: таблицы, управляющие и обрабатывающие запросы, формы, отчеты, страницы, макросы, модули.

3. Средства создания и модификации объектов базы данных.

**2.8.3 Описание (ход) работы:**

База данных представляет собой совокупность специальным образом организованных данных, хранимых в памяти вычислительной системы и отображающих состояние объектов и их взаимосвязей в рассматриваемой предметной области.

Следует учесть, что это определение не является единственно возможным. Информатика в отношении определений чаще всего не похожа на математику с ее полной однозначностью. Если подойти к понятию “база данных” с чисто пользовательской точки зрения, то возникает другое определение: база данных – совокупность хранимых операционных данных некоторого предприятия.

В базе данных предприятия, например, может храниться:

- вся информация о штатном расписании, о рабочих и служащих предприятия;
- сведения о материальных ценностях;
- данные о поступлении сырья и комплектующих;
- сведения о запасах на складах;
- данные о выпуске готовой продукции;
- приказы и распоряжения дирекции и т.п.

Даже небольшие изменения какой-либо информации могут приводить к значительным изменениям в разных других местах.

Пример. Издание приказа о повышении в должности одного работника приводит к изменениям не только в личном деле работника, но и к изменениям в списках подразделения, в котором он работает, в ведомостях на зарплату, в графике отпусков и т.п.

Поскольку основу любой базы данных составляет информационная структура, базы данных делят на три типа: табличные (реляционные), сетевые, иерархические.

Опыт использования баз данных позволяет выделить общий набор их рабочих характеристик:

- полнота – чем полнее база данных, тем вероятнее, что она содержит нужную информацию (однако, не должно быть избыточной информации);
- правильная организация – чем лучше структурирована база данных, тем легче в ней найти необходимые сведения;
- актуальность – любая база данных может быть точной и полной, если она постоянно обновляется, т.е. необходимо, чтобы база данных в каждый момент времени полностью соответствовала состоянию отображаемого ею объекта;
- удобство для использования – база данных должна быть проста и удобна в использовании и иметь развитые методы доступа к любой части информации.

Надо отметить, что база данных – это, собственно, хранилище информации и не более того. Однако, работа с базами данных трудоемкая и утомительная. Для создания, ведения и осуществления возможности коллективного пользования базами данных используются программные средства, называемые системами управления базами данных (СУБД).

Система управления базами данных (СУБД) – это система программного обеспечения, позволяющая обрабатывать обращения к базе данных, поступающие от прикладных программ конечных пользователей. Иными словами, СУБД является интерфейсом между базой данных и прикладными задачами.

Системы управления базами данных позволяют объединять большие объемы информации и обрабатывать их, сортировать, делать выборки по определённым критериям и т.п.

Основные функции СУБД – это:

- определение данных;
- обработка данных;
- управление данными.

Современные СУБД дают возможность включать в них не только текстовую и графическую информацию, но и звуковые фрагменты и даже видеоклипы.

Простота использования СУБД позволяет создавать новые базы данных, не прибегая к программированию, а пользуясь только встроенными функциями.

СУБД обеспечивают правильность, полноту и непротиворечивость данных, а также удобный доступ к ним.

Для менее сложных применений вместо СУБД используются информационно-поисковые системы (ИПС), которые выполняют следующие функции:

- хранение большого объема информации;
- быстрый поиск требуемой информации;
- добавление, удаление и изменение хранимой информации;
- вывод ее в удобном для человека виде.

В информационных системах, которые работают на ПК, совместимых с IBM PC, большое распространение получили так называемые dBASE-подобные системы управления базами данных (СУБД). Известно по крайней мере три семейства таких СУБД (dBASE, FoxPro и Clipper), однако версий оригинальных систем и их адаптированных вариантов гораздо больше. Для пользователей существенным является то, что отличаясь между собой командными языками и форматом индексных файлов, все эти СУБД используют одни и те же оперативные файлы с расширением. DBF, формат которых стал на некоторое время своеобразным стандартом баз данных.

В dBASE-подобных БД фактически использован реляционный подход к организации данных, т.е. каждый файл. DBF представляет собой двумерную таблицу, которая состоит из фиксированного числа столбцов и переменного числа строк (записей). В терминах, принятых в технической документации, каждому столбцу соответствует поле одного из пяти типов (N – числовое. C – символьное, D – дата, L – логическое. M – примечание), а каждой строке – запись фиксированной длины, состоящая из фиксированного числа полей. С помощью командных языков этих СУБД мы создаем и исправляем макеты файлов. DBF (описания таблиц), создаем индексные файлы, пишем пиктограммы работы с базами данных (чтение, поиск, модификация данных, составление отчетов и многое другое). Характерной особенностью файла DBF является простота и наглядность: физическое представление данных на диске в точности соответствует представлению таблицы на бумаге.

Однако в целом системы, построенные на основе файлов DBF, следует считать устаревшими. Многие механизмы реляционных БД, рассмотренные выше, в dBASE-подобных системах либо не поддерживаются, либо создаются пользователями и программистами «кустарным» способом.

Большую популярность до сего времени имеют и другие СУБД (с другим форматом файлов) – Paradox, Clarion, db_Vista и тд. Следует подчеркнуть, что перечисленные системы ведут родословную от MS-DOS, однако ныне почти все они усовершенствованы и имеют версии для Windows.

Среди современных реляционных систем наиболее популярны СУБД для Windows – Access фирмы Microsoft, Approach фирмы Lotus, Paradox фирмы Borland. Многие из этих систем поддерживают технологию OLE и могут манипулировать не только числовой и текстовой информацией, но и графическими образцами (рисунками, фотографиями) и даже звуковыми фрагментами и видеоклипами.

Перечисленные СУБД часто называют настольными, имея в виду сравнительно небольшой объем данных, обслуживаемых этими системами. Однако с ними часто работают не только индивидуальные пользователи, но и целые коллективы.

Вместе с тем, в центр современной информационной технологии постепенно перемещаются более мощные реляционные СУБД с так называемыми SQL-доступом (SQL – это язык запросов). В основе этих СУБД лежит так называемая технология «клиент-сервис». Среди ведущих производителей таких систем – фирмы Oracle, Centura (Gupta), Sybase, Informix, Microsoft и другие. Появились также объектные и объектно-реляционные СУБД.

В последнее время стали среди СУБД наиболее популярными и используемые в практике Access, Lotus, Oracle.

Разберем наиболее используемую программу Access.

Access – в переводе с английского означает “доступ”. MS Access – это функционально полная реляционная СУБД. Кроме того, MS Access одна из самых мощных, гибких и простых в использовании СУБД. В ней можно создавать большинство приложений, не написав ни единой строки программы, но если нужно создать нечто очень сложное, то на этот случай MS Access предоставляет мощный язык программирования – VisualBasicApplication.

Популярность СУБД Microsoft Access обусловлена следующими причинами:

- Access является одной из самых легкодоступных и понятных систем как для профессионалов, так и для начинающих пользователей, позволяющая быстро освоить основные принципы работы с базами данных;
- система имеет полностью русифицированную версию;
- полная интегрированность с пакетами Microsoft Office: Word, Excel, Power Point, Mail;
- идеология Windows позволяет представлять информацию красочно и наглядно;
- возможность использования OLE технологии, что позволяет установить связь с объектами другого приложения или внедрить какие-либо объекты в базу данных Access;

- технология WYSIWIG позволяет пользователю постоянно видеть все результаты своих действий;
- широко и наглядно представлена справочная система;
- существует набор “мастеров” по разработке объектов, облегчающий создание таблиц, форм и отчетов.

После запуска системы появится главное окно Access (рис. 1). Здесь можно открывать другие окна, каждое из которых по-своему представляет обрабатываемые данные. Ниже приведены основные элементы главного окна Access, о которых необходимо иметь представление.

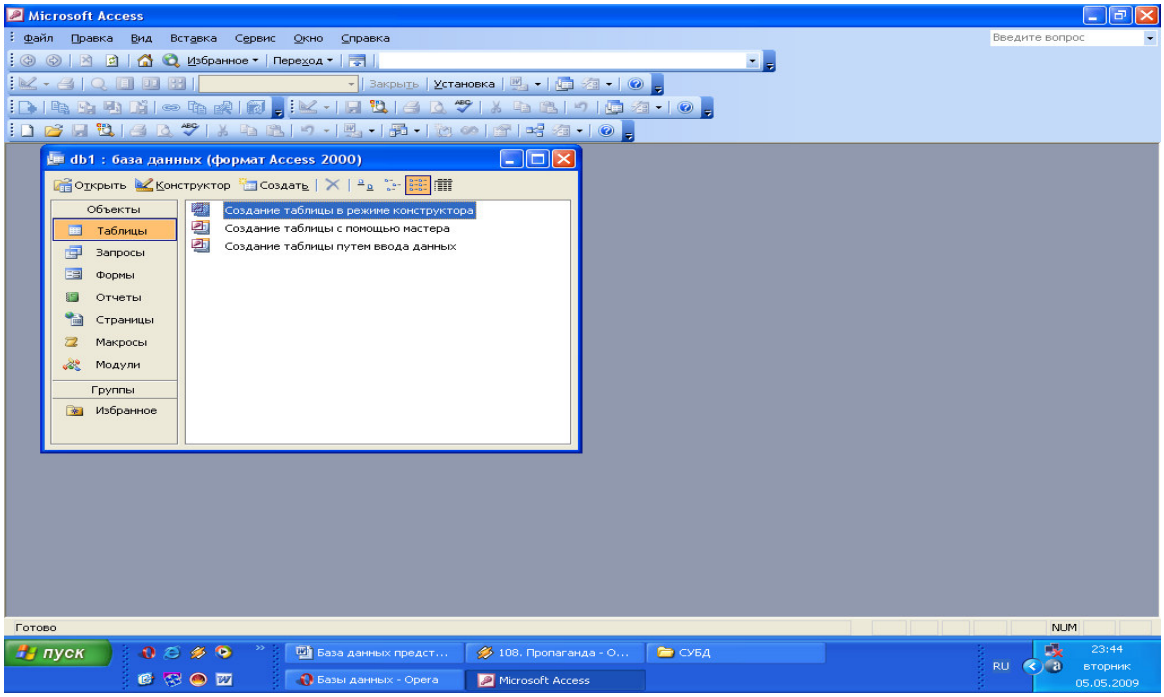



Рис.1. Экран СУБД Access.

В строке заголовка отображается имя активной в данный момент программы. Строка заголовка главного окна Access всегда отображает имя программы MICROSOFT Access.

Пиктограмма системного меню – условная кнопка в верхнем левом углу главного окна практически любого приложения. После щелчка на этой пиктограмме появляется меню, которое позволяет перемещать, разворачивать, сворачивать или закрывать окно текущего приложения и изменять его размеры. При двойном щелчке на пиктограмме системного меню работа приложения завершается.

Панель инструментов – это группа пиктограмм, расположенных непосредственно под полосой меню. Главное ее назначение – ускоренный вызов команд меню. Кнопки панели инструментов тоже могут изменяться в зависимости от выполняемых операций. Можно изменять размер панели инструментов и передвигать ее по экрану. Также можно

отобразить, спрятать, создать новую панель инструментов или настроить любую панель инструментов.

Окно базы данных появляется при открытой базе данных. В нем сосредоточены все “рычаги управления” базой данных. Окно базы данных используется для открытия объектов, содержащихся в базе данных, таких как таблицы, запросы, отчеты, формы, макросы и модули. Кроме того, в строке заголовка окна базы данных всегда отображается имя открытой базы данных.

С помощью вкладки объектов можно выбрать тип нужного объекта (таблицу, запрос, отчет, форму, макрос, модуль). Необходимо сказать, что при открытии окна базы данных всегда активизируется вкладка-таблица и выводится список доступных таблиц базы данных. Для выбора вкладки других объектов базы данных нужно щелкнуть по ней мышью.

К основным объектам Access относятся таблицы, запросы, формы, отчеты, макросы и модули.

Таблица – это объект, который определяется и используется для хранения данных. Каждая таблица включает информацию об объекте определенного типа. Как уже известно, таблица содержит поля (столбцы) и записи (строки). Работать с таблицей можно в двух основных режимах: в режиме конструктора и в режиме таблицы.

Запрос – это объект, который позволяет пользователю получить нужные данные из одной или нескольких таблиц. Можно создать запросы на выбор, обновление, удаление или на добавление данных. С помощью запросов можно создавать новые таблицы, используя данные уже существующих одной или нескольких таблиц.

Форма – это объект, в основном, предназначенный для удобного ввода отображения данных. Надо отметить, что в отличие от таблиц, в формах не содержится информации баз данных (как это может показаться на первый взгляд). Форма – это всего лишь формат (бланк) показа данных на экране компьютера. Формы могут строиться только на основе таблиц или запросов. Построение форм на основе запросов позволяет представлять в них информацию из нескольких таблиц.

Отчет – это объект, предназначенный для создания документа, который впоследствии может быть распечатан или включен в документ другого приложения. Отчеты, как и формы, могут создаваться на основе запросов и таблиц, но не позволяют вводить данные.

Макрос – это объект, представляющий собой структурированное описание одного или нескольких действий, которые должен выполнить Access в ответ на определенное событие. Например, можно определить макрос, который в ответ на выбор некоторого

элемента в основной форме открывает другую форму. С помощью другого макроса можно осуществлять проверку значения некоторого поля при изменении его содержания. В макрос можно включить дополнительные условия для выполнения или невыполнения тех или иных включенных в него действии.

Работа с формами и отчетами существенно облегчается за счет использования макрокоманд. В MS Access имеется свыше 40 макрокоманд, которые можно включать в макросы. Макрокоманды выполняют такие действия, как открытие таблиц и форм, выполнение запросов, запуск других макросов, выбор опций из меню, изменение размеров открытых окон и т.п. Макрокоманды позволяют нажатием одной (или нескольких одновременно) кнопки выполнять комплекс действий, который часто приходится выполнять в течение работы. С их помощью можно даже осуществлять запуск приложений, поддерживающих динамический обмен данных (DDE), например MS Excel, и производить обмен данными между вашей базой данных и этими приложениями. Один макрос может содержать несколько макрокоманд. Можно также задать условия выполнения отдельных макрокоманд или их набора.

Модуль – объект, содержащий программы на MS Access Basic, которые позволяют разбить процесс на более мелкие действия и обнаружить те ошибки, которые невозможно было бы найти с использованием макросов.

Завершив работу с Access (или с ее приложением), надо корректно закончить сеанс. Простое выключение компьютера - плохой метод, который может привести к возникновению проблем. При работе WINDOWS приложения используют множество файлов, о существовании которых пользователь может даже не подозревать. После выключения машины эти файлы останутся открытыми, что в будущем может сказаться на надежности файловой системы жесткого диска.